International Journal of Industrial Engineering & Production Research (2016)

June 2016, Volume 27, Number 2
pp. 167-178

http://IJIEPR.iust.ac.ir/

A Fixed and Flexible Maintenance Operations Planning in a Parallel Batch Machines Production System

Saber Molla-Alizadeh Zavardehi[1], Amir Noroozi[2,*] & Hadi Mokhtari[3]

Saber Molla-Alizadeh-Zavardehi, *Department of Industrial Engineering, Islamic Azad University, Masjed Soleyman Branch*
Amir Noroozi, *PhD Candidate, Department of Industrial Engineering, Iran University of Science and Technology*
Hadi Mokhtari, *Assistant Professor, Department of Industrial Engineering, Faculty of Engineering, University of Kashan*

KEYWORDS

Manufacturing systems,
Preventive maintenance,
Total maintenance program,
Batch processing

ABSTRACT

In today's real-world manufacturing systems, the importance of an efficient maintenance schedule program cannot be ignored, because it plays an important role in the success of manufacturing facilities. However, most manufacturers suffer from lack of a total maintenance plan for their crucial manufacturing systems. Hence, in this paper, we study a maintenance operations planning optimization on a realistic variant of parallel batch machines manufacturing system which considers non-identical parallel processing machines with non-identical job sizes and fixed/flexible maintenance operations. To reach an appropriate maintenance schedule, we propose a solution frameworks based on an Artificial Immune Algorithm (AIA). Then, we introduce a new method to calculate the affinity value by using an adjustment rate. Finally, the performance of the proposed methods are investigated. Computational experiments, for a wide range of test problems, are carried out in order to evaluate the performance of methods.

© 2016 IUST Publication, IJIEPR. Vol. 27, No.2, All Rights Reserved

1. Introduction

The true cost of a breakdown is sometimes difficult to assess for a machine. It is obvious that this cost is more than the cost of labor and materials of preventive services. When the breakdown incurs a stop to production, the cost is significantly high because no products are being produced. In order to maintain machines and equipment in an appropriate operating condition, manufacturers need to preventively maintain all their facilities. This paper deals with a problem of scheduling jobs on non-identical parallel batch machines with maintenance considerations. A batch machine (BM) is one that is capable of simultaneously processing a group of jobs, such that processing time of all jobs in a batch is the same. The characteristics of each job is two-fold:

* **Corresponding author: Amir Noroozi**
Email: *amir_noroozie@iust.ac.ir*
Received 9 June 2015; revised 13 June 2016; accepted 22 June 2016

(i) job size which is no greater than batch size *B* and (ii) job processing time. There are two important decisions should be made on BM problems: (i) grouping jobs into batches; (ii) scheduling the established batches to satisfy an objective measure. Examples of BP manufacturing system occur in semiconductor industries, steel foundry and chemical processes, electronics manufacturing, environmental stress - screening chambers, wafer fabrication, food, mineral, pharmaceutical and construction industries, etc. In the sequel, we discuss the literature on parallel BMs related to the current work. Chang *et al.* (2004) designed a simulated annealing algorithm to reach the best value of system makespan on parallel BMs. An enumerative function of both problem size and the least number of required batches was devised for such a problem by Husseinzadeh Kashan et al. (2008). Then, a hybrid genetic heuristic was suggested minimizing maximum completion time. In addition, a bi-objective problem was considered by Yazdani Sabouni et al. (2008) in order to find the best amount of total completion time and maximum lateness measures. Moreover, a constructive heuristic algorithm presented by Damodaran and Velez-Gallego (2009) to optimize the makespan and a Greedy Randomized Adaptive Search Procedure (GRASP) approach were designed in another research by Damodaran *et al.* (2009).

Motivated by a scheduling problem in semi-conductor wafer fabrication, Mönch et al. (2005) discussed two different approaches for scheduling jobs on parallel batch machines with incompatible families and unequal readiness. Liu et al. (2009) considered a problem in a burn-in operations of the final testing stage of semi-conductor manufacturing. The strong NP-hardness of the problem, where the jobs have deadlines, was discussed. And, a polynomial time approximation scheme was devised with the aim of minimizing the maximum lateness. A dynamic job arrivals case of problem was presented by Malve and Uzsoy (2007), where two versions of the Release Date Update heuristic were discussed and a genetic algorithm was proposed incorporating heuristics. For the same problem, Wang and Chou (2010) constructed a mixed integer programming model, a genetic and a simulated annealing algorithm to minimize the makespan of system. Additionally, Damodaran and Vélez-Gallego (2012) proposed a simulated annealing algorithm and a lower bound procedure. Moreover, we can find research findings in literature considering the planning of maintenance in manufacturing and production plants (Jamshidi and Seyyed Esfahani, 2015; Jalali Naini et al., 2009; Riahi and Ansarifard, 2008; Mokhtari et al., 2012; Mokhtari and Dadgar, 2015). As can be seen, all of the above pieces of research assumed that BMs are continuously available during the entire horizon. However, in real-world situations, machines are subject to unpredicted environmental and technical circumstances; therefore, they may not work efficiently and profitably without a scheduled preventive maintenance (PM) program. In this paper, contrary to previous BMs pieces of research, in order to address a real scheduling scheme closer to the real situations, a realistic variant of parallel BMs scheduling is proposed and discussed where scheduling of PM operations is also considered in production scheduling.

Since the problem under consideration is NP-hard in a strong sense, an Artificial Immune Algorithm, as a novel efficient metaheuristic, is devised for the problem in the current paper. This is due to simplicity, easy implementation, fast convergence, and robustness of AIA. Besides, it is known that the quality of the solutions obtained by a metaheuristic algorithm is strongly affecting the different levels of the parameters. Consequently, the Taguchi experimental design is also employed as a parameter tuning method to calibrate the used parameters. Details of the problem statement and solution technique will be presented in subsequent sections.

2. Notations and Problem definition

In the following, we first discuss maintenance and PM operations, and then present BM scheduling considering PM. Due to the previous research pieces in literature, there are commonly two types of PM, i.e., *fixed PM* and *Non-fixed PM*, which will be described in the following subsections.

2-1. Fixed PM

The *fixed PM* is one for which PM operations are planned in advance, i.e., the starting times of maintenance operations are fixed beforehand. More precisely, in this type of PM, the fixed time periods (T_{FPM}) are predefined without considering probabilistic models, and maintenance operations are performed exactly in these periods.

2-2. Non-fixed PM

The *Non-fixed PM* is one that the schedule of preventive maintenance periods are delineated

jointly with the schedule of jobs. In other words, the starting times of the PM operations are flexible. In *Non-fixed PM*, the time of the PM operations is not determined precisely, but the time needed to perform PM activities is increasing, instead of being fixed. After that, machine works continuously for this computed period of time, and the PM operation should be done. Details of this PM approach will be discussed more in the sequel. To compute optimal PM periods in *Non-fixed PM* approach, there are various policies in literature. We employed two models of the most important policies in the current work: (i) computing optimal periods based on maximizing the machines' availability (*Non-fixed PM.i*); (ii) computing optimal periods based on keeping a minimum reliability threshold for a given production period t (*Non-fixed PM.ii*). In our approach, optimal PM periods are determined considering probabilistic models and perform according to these periods. Due to flexibility of the Weibull distribution to determine the time of equipment failure with variable failure rates, this model is one of the most commonly used models. At a Weibull probability distribution, $T\sim W[\theta, \beta]$ with $\beta>1$. where β is shape parameter and θ is scale parameter.

(i) *Non-fixed PM.i*: since the machine is unavailable during maintenance period, an optimal PM interval has been determined for a manufacturing environment by maximizing its constraint availability. Let T_{PMop} be the optimal interval between two sequential PM operations. Since the time to failure follows a Weibull probability distribution, $T\sim W[\theta, \beta]$ with $\beta>1$, according to Cassady and Kutanoglu (2003), the optimal maintenance interval T_{PMop} can be obtained as follows:

$$T_{PMop} = \theta \cdot \left[\frac{t_p}{t_r(\beta - 1)} \right]^{1/\beta} \quad (1)$$

where t_r represents the number of time units that repair takes and t_p denotes the number of time units of the PM operations. As an example, let time to failure follow a Weibull distribution with θ=290, β=2 together with t_r =8^h and t_p =1^h. Then, by employing Eq. (1), the optimal PM interval T_{PMop} is calculated as 102.53^h.

(ii) *Non-fixed PM.ii*: in an unreliable manufacturing system, failure rate decreases with time; hence, it may be affected by failures. The *Non-fixed PM.ii* approach consists of implementing a systematic PM after a time T_{PM} to ensure a minimum reliability ($R_0(t)$) of the system. It is supposed that PM operations will be carried out at intervals 0, $1T_{PM}$, $2T_{PM}$, $3T_{PM}$, ..., nT_{PM} where components reach the as-good-as-new condition. If the time to failure follows a Weibull probability distribution, $T\sim W[\theta,\beta]$ with $\beta>1$ (Ruiz *et al.* 2007), the time between PMs in *Non-fixed PM.ii* can be calculated by:

$$T_{PM} = \left[-\theta^{\beta} \frac{\ln R_0(t)}{t} \right]^{1/(\beta-1)} \quad (2)$$

As an example, let us consider that the time to failure of a machine is characterized with parameters θ=290 and β=2. The aim is to delineate the time between maintenance operations in such a way that reliability is 95% ($R_0(t)$=0 95%). To this end, by employing Eq. (2), we obtain T_{PM}=25.8^h.

In the sequel, before presenting the BM scheduling considering PM, we first express First-First (FF) heuristic based on Melouk *et al*. (2004) for grouping jobs into batches.

2-3. First - first heuristic

The first-first (FF) heuristic is a methodology where batches are formed first and then sequenced. Using this heuristic, the first job in the list is placed in *Batch 1*. When we intend to place job j in a batch, it is placed in the lowest indexed batch if the job size does not exceed the remaining capacity of the batch (i.e., $s_j \leq S - \sum_{k\in J, k\neq j} s_j X_{kb}$). This process repeats itself until all jobs are allocated to batches. Besides, the sequence of batches is based on the batch number, i.e., *Batch* 1*, Batch* 2*, … , Batch B*. It is obvious that the maximum possible number of batches will be equal to the number of jobs.

2-4. BM Scheduling considering PM

As mentioned before, a few works have been presented so far in which PM operations are incorporated with parallel machine scheduling problem. The aim of this section is to give a detailed description on the parallel BM scheduling problem with PM operations studied in this paper. This is classified into four phases as follows.

***Phase* 1.** Determination of PM intervals. As mentioned above, in this paper, two types of PM are considered: *fixed PM* and *Non-fixed PM,* in which two approaches are introduced in a *Non - fixed PM*. In this phase, industry environment can carry out one of the approaches based on its policy, consequently, determine PM intervals.

How to compute PM intervals in *fixed PM* due to characteristics kind of BM or other conditions can be determined simply. Computing PM intervals in *Non-fixed PM.i* and *Non-fixed PM.ii* has been described in detail in pervious sections.

***Phase* 2.** Forming batches.

***Phase* 3.** Sequencing formed batches on parallel BMs.

To perform two phases 2 and 3, we use FF methodology that is detailed above. These two phases are presented in the example of Fig.1. Based on FF, in Batch 3, difference between processing time of J_{10} (as processing time of Batch 3 e.g. P_3) and J_5 is equal to 4, which is lower than ξ. But, since summation size of job J_{10} and J_5 is 11 and it does exceed B, the difference between processing time of J_{10} and next job in the list, e.g., J_6, is computed, which is lower than ξ. Moreover, summation size of jobs J_{10} and J_6 is 3 and do not exceed B, hence the need for inserting Batch 3 after J_{10}, J_6. Then, since 7 units of batch sizes are empty, difference between P_3 and next job, e.g., J_2, in sequence is computed. But, it is not lower than ξ here; and hence, can be inserted in Batch 3. Grouping other jobs in batches is the same. As mentioned above, FF automatically sequenced batches based on LPT.

Phase 4. Scheduling (batch processing and carrying out PMs on BMs). Now, each batch should be processed on each BM. In this phase, PM operations are embedded in BM scheduling. The interruption of batches is not allowed once processing is started, i.e., the batches are non-preemptive and resumable. In order to decide on production sequence, defining a criterion is necessary to schedule PM operations. Ruiz *et al.* (2007) introduced a simple criterion for *Non-fixed PM* with good results in many situations. We employed this criterion for our problem with some modifications as follows. Whenever there is overlap between a batch and PM operations, the PM operation is processed at first, and the processing of batch is postponed until PM operation terminates. When a new batch is to be processed on a BM, the total accumulated processing time will be computed. Now, if shop planning is based on *fixed PM*, then accumulated processing time will be compared with PM interval. And, if shop planning is based on *Non-fixed PM*, then accumulated processing time will be compared with T_{PMop} or T_{PM}. In this comparison, if the accumulated processing time is longer than PM interval in *Fixed PM*, and T_{PMop} in *Non-fixed PM.i* or T_{PM} in *Non-fixed PM.ii*, the PM operation is processed at first and the process of the next batch is postponed. Let us consider an example explained in Figure 1. In this example, we have four batches with sequence {B1, B2, B3, B4} which should be processed on two BMs available. As can be seen, without considering PM operations, C_{max} is calculated as 34.

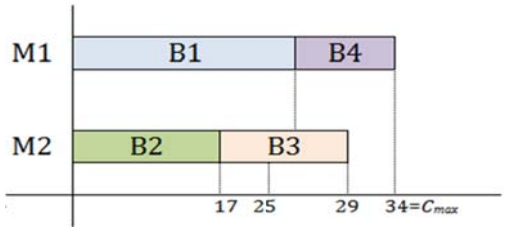

Fig. 1. Gantt chart of the solution for the example

Let us suppose that PM activities are scheduled based on *fixed PM* with T_{PMop}=40 for machine M1 and T_{PMop}=25 for machine M2. The duration of PM activities is set to be 10 (D_{PM}=10) and 8 (D_{PM}=8) on machines M1 and M2, respectively. As it is clear from Figure 2, processing of B1 lasts 25 time units on M1. At this stage, it is not possible to process B4, because it has processing time of 9, which would finish at the time of 34 while T_{FPM} = 30. Therefore, machine M1 would be left idle for 5 time units, and then PM begins at moment 30, and lasts for 10 time units. After performing PM operation, B4 is processed. This approach is repeated on machine M2 with the same manner.

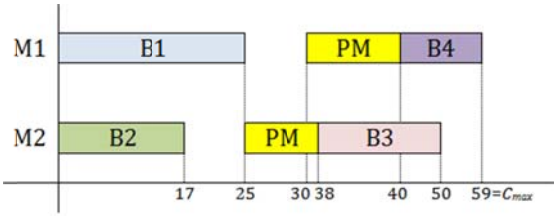

Fig. 2. Gantt chart after applying the *fixed PM*

Now, consider another case with T_{PMop}=30 for machine M1 and T_{PMop}=25 for machine M2. The durations of PM on machines M1 and M2 are assumed to be 10 and 8, respectively. PM scheduling is based on *Non-fixed PM.i*. in this case. It is also assumed that shop has the same batches with the same sequence in the current horizon. After B1, B4 has to be processed without any wait. It has the processing time of 9, which would result in an accumulated total processing time of 34 greater than T_{PMop}=30. Hence, PM should be performed at first, and the process of B4 should be postponed. After performing the first PM with duration of 10 that lasts from 25 to 35, the accumulated total

processing time restarts from zero. Then, processing of other remaining batches begins. For machine M2, the same procedure will be done. Figure 3 depicts Gantt chart of example with *Non-fixed PM.i*.

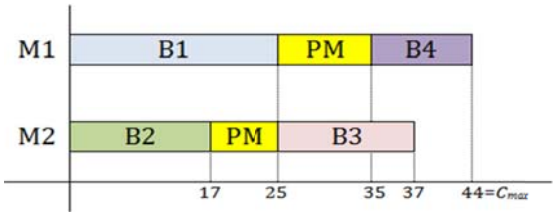

Fig. 3. Gantt chart after applying the flexible *PM.i*

Carrying out PM with *Non-fixed PM.ii* case reveals the same results as *Non-fixed PM.ii* case; however, T_{PMOP} is first calculated by Eq. (2), and then the rest of steps should be done according to their order. In other words, each BM will be exactly in operation T_{PMop} time units if we apply *Non-fixed PM.i* (Eq. (1)), or if *Non-fixed PM.ii* is applied (Eq. (2)) after T_{PM} time units in operation. As we can see, the simplicity of the approach and simple coding allows for implementing the scheduling of PM operations with ease. Therefore, this approach is adaptable and extendable to any BM scheduling problem. It is notable that C_{max} is 59 *in fixed PM*, while it is 44 in *Non-fixed PM.i*.

3. Methodologies

The use of classical optimization techniques is limited, because there are a large number of variables and constraints in real-world situations, and hence complexity of problem is high. In such cases, metaheuristics are appropriate for solving alternative. Therefore, in the current paper, an artificial immune algorithm as a new efficient metaheuristic, is proposed to solve the presented model.

3-1. An artificial immune algorithm

The natural immune system is an adaptive pattern recognition system defending the body from foreign pathogens like bacteria and viruses. This system is able to categorize all cells within the body as either those belonging to its own kind or those that have a foreign origin (Dasgupta, and Gonzalez, 2003). Detection is complicated as pathogens can evolve rapidly, producing adaptations that avoid the immune system and allow the pathogens to successfully infect hosts. The immune system learns to distinguish between its own cells and malefic external invaders in order to be protective. This process is known as discrimination. Vertebrates have an adaptive immune system and are capable of learning to recognize and eliminate new antigens. In biological systems, an immune response is based on two types of lymphocytes in body, i.e., T-cell originating in the thymus gland and B-cell originating in bone marrow (de Castro and Timmis, 2002a and 2002b). The role of these receptors on the surface of the lymphocytes is to recognize and bind the antigen.

Once a B-cell is sufficiently stimulated through close affinity to a presented antigen, it rapidly produces clones of itself. At the same time, it produces mutations at particular sites in its gene which enable the new cells to match the antigen more closely. There is a very rapid proliferation of immune cells, successive generations of which are better and better matches for the antigens of the invading pathogen. B-cells which are not stimulated eventually die because they do not match any antigen in the body. Biologically, immune systems generate a set of antibodies to recognize foreign substances, known as antigens and, to eliminate them. The mechanisms are able to recognize, in which antibodies are better at eliminating the antigens, and produce more variations of those antibodies in the next generation of antibodies. Each antibody is assigned a value, called affinity, showing the capability of the antibody to eliminate antigens. The objective functions and constraints are represented as antigen inputs, while the solution process is simulated by antibody production in the feasible space. The affinity in the artificial immune algorithm (AIA) is equivalent to the objective function. The calculation of the affinity between antibodies is embedded within the algorithm to determine the promotion/suppression of antibody production.

Three commonly applied types of AIAs are immune network algorithm (INA), negative selection algorithm (NSA), and clonal selection algorithm (CSA). The main operators of the CSA are cloning selection and affinity maturation. The affinity and clonal selection maturation principles are used to explain how the immune system reacts to pathogens, and how it improves its capability of recognizing and eliminating. When an antigen is detected, those antibodies that best recognize this antigen will proliferate by cloning. This process is called a clonal selection principle. Affinity maturation is the whole mutation processes and selection of the variant offspring that better

recognizes the antigen (de Castro and Timmis, 2003). Two basic mechanisms of affinity maturation are hypermutation and receptor editing (de Castro and Von Zuben, 2002c). Hypermutation is in a similar way with mutation in genetic algorithms. It ensures large mutations for low-affinity cells in order to obtain cells with a higher affinity. Its procedure is handled by receptor editing. The natural IA is a very complex system with several mechanisms to defend against pathogenic organisms. However, the natural immune system is also a source of inspiration for solving optimization problems.

3-2. Antibody representation and initialization

In this section, we use string of discrete number representation as applied in the proposed genetic algorithm. For the problem with n customer jobs, the representation is a permutation of n digits in length, in which all digits are between 1 and n.

3-3. Fitness evaluation

The higher fitness value (FV) means the better antibody. So, we define the following function to evaluate the fitness of each antibody by:

$$\text{Fitness value}(p) = \frac{1}{\text{Objective value}(p)}$$

Thus, the lower the objective function is, the higher the affinity value becomes.

3-4. The best selection

In this section, the best known antibody based on affinity is found.

3-5. Similarity evaluation

The similarity between each antibody with the best known antibody obtained so far is evaluated. All the remaining antibodies will be compared with the best antibody one by one to count the number of different chromosome positions. This evaluation is based on the similarity rate (Sr) computed by: $SR = \frac{k}{n}$

where k is the number of digits that are the same as those in the best antibody, and n is the length of a string number.

3-6. Affinity value (AV) calculation

The affinity evaluation of antibodies is an important index for the AIA during optimization process. Since a good candidate antibody may be able to multiply and dominate the mating pool prematurely and may restrict the search space to candidate antibodies, we use the following method to prevent good candidate antibody from becoming overly dominant. To assist the search, the proposed AIA uses fitness value (FV), similarity rate (SR), and adjustment rate (AR) to compute the affinity value by:

$$\text{Affinity value}(p) = (1 - SR \times AR) \times (FV(p))$$

where AR is a parameter that has influence on the impact of SR on the affinity value. The higher AR gets, the lower coefficient (i.e., $1-SR \times AR$) becomes. To maintain diversity, each antibody that has higher SR is assigned a lower coefficient (i.e., $1-SR \times AR$) multiplied by FV in order to decrease FV more than other antibody's reduction.

3-7. Mating pool expansion

popsize-1 antibodies from the full population, including the best antibody, are selected with replacement and according to their fitness. Those antibodies, which have higher fitness value, are selected more often.

3-8. Crossover

We select *popsize*×*pc* pairs of parents from the mating pool and perform crossover on the parents at random. We use some crossover operators applied to the proposed GA.

3-9. Hyper mutation

We select the remaining antibodies from mating pool and mutate the individual bits. Engin and Doyen (2004) used a two-phase mutation procedure. At first, the selected clones suffer a mutation operator. If the objective value of the mutated clone is better than that of the original clone, then the mutated one is stored in the place of the original one. Otherwise, the clone will be mutated again with another mutation operator. Due to having four mutation operators, as described in the proposed GA, there are 12 (i.e., 4×3) permutations.

3-10. Receptor editing

The worst $\alpha\%$ of the whole population in the antibody population is eliminated and replaced with randomly created antibodies. The overall procedure is shown in Figure 4.

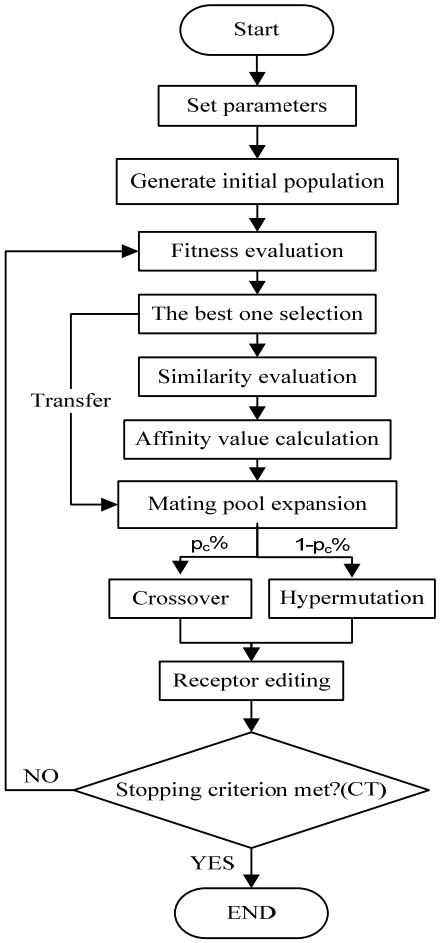

Fig. 4. Overall proposed procedure

4. Experiments and Results

As mentioned earlier, genetic algorithm (GA) is the most common metaheuristic employed to solve the batch machines problem in literature. Therefore, we employed GA as a benchmark for investigating AIA suggested in the current research. GAs attempt to mimic the biological evolution process for discovering good solutions. They are based on a direct analogy to the Darwinian natural selection and mutations in biological reproduction and belong to a category of heuristics known as randomized heuristics that employ randomized choice operators in their search. Nowadays, GA is considered to be one of the typical meta-heuristic approaches for tackling both discrete and continuous optimization problems. Theoretical analyses suggest that GAs can quickly locate high performance regions in extremely large and complex search spaces. GAs work by generating chromosomes (e.g., numeric vectors), representing a possible solution to a problem. The individual components (e.g., numeric values) within a chromosome are called genes. Chromosomes are then evaluated according to a fitness (or objective) function. A fitness value is assigned to each chromosome according to its performance. The more desirable the chromosome is, the larger the fitness value becomes. A typical iteration of a GA, called generation, proceeds as follows. The best chromosomes of the current population are copied directly to the next generation (i.e., reproduction). A selection mechanism chooses chromosomes of the current population in such a way that the chromosome with the higher fitness value has a greater probability of being selected. The selected chromosomes mate and generate new offspring by "crossover" (i.e., the probabilistic exchange of values between vectors). After the mating process, each offspring might mutate by another mechanism, called "mutation" (i.e., the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. The GA's search process typically continues until a pre-specified fitness value is reached; a set amount of computing time passes or until no significant improvement occurs in the population for a given number of iterations. The key to finding a good solution using a GA lies in developing a good chromosome representation of solutions to the problem.

Most of the evolutionary algorithms use a random procedure to generate an initial set of solutions. The initialization of a chromosome presented in a string of discrete number is performed from randomly generated n digits in range $[1, n]$.

Since the objective is the minimization of the total cost, better solutions are those results in a lower objective function. The higher fitness value means the better chromosome. So, we define the following function to calculate each fitness value:

$$\text{Fitness Value} = \frac{1}{\text{Objective Function}}$$

Using the roulette-wheel selection mechanism, the higher fitness value a solution has, the more chance it has to be selected.

4-1. Genetic operators

4-1-1. Reproduction

With more probability, better parents can generate better offspring. So, it is necessary to transfer the best solutions of each generation to the next. Therefore, chromosomes with higher fitness values are more desirable, so p_r% of the chromosomes with the greater fitness values is

automatically copied to the next generation, in which this is called the elite strategy.

4-1-2. Crossover

Crossover operates on two chromosomes at a time and generates offspring by combining both chromosomes features. The main purpose of this operator is to generate 'better' offspring, i.e., to create better sequences after combining the parents. As we assign p_r of the chromosomes in a generation to reproduction, $1–p_r$ remaining chromosomes are generated through the crossover operator.

One-point crossover:

Input: Two chromosomes as parents

Output: Two feasible chromosomes as offspring

Step 1: One point is randomly selected to divide both selected parents into two separate parts. If n is the number of genes in a chromosome, there are n–1 crossover points. One of these points is selected with the equal probability. The digits on the first side of Parents I and II are inherited by the Offsprings I and II, respectively.

Step 2: Transfer digits from the second side of Parent I to the second side of Offspring II in a left to right order. Similarly, transfer digits from the second side of Parent II to the second side of Offspring I.

Step 3: Transfer those digits from the second side of Parent I, which cannot be copied to the second side of Offspring II, to remaining empty positions of the second side of Offspring I. Do similarly for remaining empty positions of Offspring II.

Two-point crossover:

Input: Two chromosomes as parents

Output: Two feasible chromosomes as offspring

Step 1: Two points are randomly selected to divide both selected parents into three separate parts. The digits on the middle side of Parents I and II are inherited by the middle side of Offsprings I and II, respectively.

Step 2: Transfer digits from the first and third sides of Parent I to the first and third sides of Offspring II in a left to right order. Similarly, transfer digits from the first and third sides of Parent II to the first and third sides of Offspring I.

Step 3: Transfer those digits from the first and third sides of Parent I, which cannot be copied to the first and third sides of Offspring II, to the remaining empty positions of the first and third sides of Offspring I. Do similarly for the remaining empty positions of Offspring II.

Position-based crossover:

Input: Two chromosomes as parents

Output: Two feasible chromosomes as offspring

Step 1: A set of the same positions from each parent is selected randomly. Each position is independently marked with the probability of 0.5. Transfer the digits in these positions from Parents I and II into the corresponding positions in Offsprings I and II, respectively.

Step 2: Transfer the unmarked digits from Parent I to the corresponding positions of Offspring II in the left to the right order. The same procedure is applied to produce the first offspring by Parent II.

Step 3: Transfer those unmarked digits from Parent I, which cannot be copied to Offspring II, to the remaining empty positions of Offspring I. Do similarly for the remaining empty positions of Offspring II.

4-1-3. Mutation

The mutation operator is used to rearrange the structure of a chromosome and to slightly change the sequence, i.e., generating a new but similar sequence. This operator can also be considered as a simple form of a local search. The probability of mutating an offspring is called the probability of mutation, p_m, which is usually a small number. After generating an offspring by a crossover operator, a random number from uniform [0,1] is dedicated to the offspring. If this random number is less than or equal to p_m, then the mutation operator will be performed on that offspring. We present four mutation operators, namely swap mutation, big swap mutation, inversion mutation, and displacement mutation. In the swap mutation operator, two adjacent genes are swapped. In a big swap mutation, two genes are selected and swapped. In the inversion mutation, two positions are selected, and the sub-string is inverted between two positions. In the displacement mutation, a sub-string is selected and inserted in a position.

4-2. Data, factors and levels

Data required for testing effectiveness of our algorithms for the parallel BMs with maintenance problems includes two parts, data related to the production scheduling and data related to the PM. The first part of the required data consists of data related to the production scheduling. Test problem instances of this part are randomly generated in a manner similar to Husseinzadeh Kashan *et al.* (2008). For covering various types of problems, four factors are identified: number of jobs (n), number of BMs (m), variation in job processing times (p_j), and variation in job sizes (s_j). In general, six categories of problems are generated by combining three levels of job sizes (small, large, and combination of small and large (mixed)) and two levels of processing times. Processing times

and job sizes are generated from discrete uniform distribution. The factors, their levels and ranges, are shown in Table 1. Also, we consider the machine capacity to be 10 in all instances.

Tab. 1. Factors and levels

Factors	levels
m	2,4
n	10, 20, 50, 100, 200
p_j	Uniform [1,10] and Uniform [1,20]
s_j	Uniform [1,10], Uniform [2,4], Uniform [4,8]

The second part of data consists of shape parameter (β), scale parameter (θ), the duration of the PM operations (D_{PM}), the number of time units the repair takes (t_r), and the number of time units of the PM (t_p). These data are divided into two subparts, each of which considers one PM policy. For each configuration of n and m, β = {2, 3, 4} is defined. We assume that the duration of PM operations is uniformly distributed in a wide range of values. Because we want to consider some short maintenance actions, like cleaning, tightening of bolts or lubrication, and also some longer maintenance actions such as replacements of parts or thorough inspections, D_{PM} is defined as U[1,5], U[1,10], and U[1,15] for processing time generated from the discrete uniform [1,10] and U[1,10], U[1,20], and U[1,30] for processing time generated from the discrete uniform [1,20]. That is, there are three cases for each processing time where the average D_{PM} is 50%, 100%, or 150% of the processing times. In the case of policy II, t_p is set at 1 and t_r at 8 for all the experiments. For setting θ, the levels of θ are chosen so as to make sure that a significant number of PM operations would be carried out in each machine. It is known that a small value for θ would result in very little or even no PM operations, while a very large value would possibly impede performing certain processing of jobs on machines without interruptions due to the small amount of time between PM operations. Also, it is necessary to deal with that if the time between two consecutive PM operations is lower than the maximum processing time, some jobs could be never processed. On the other hand, if this time becomes very large, it is very likely no PM operations are required. Consequently, the levels of θ are chosen so to make sure that a significant number of PM operations would be carried out in each machine, and generating T_{PMop} must be done with the great care. Doing so, we need to define two new artificial variables "B" and "X" to estimate the workload on the batch processing machines as follows:

$$B = \frac{\sum_{j=1}^{n} s_j}{0.8 \times S}; \qquad X = \frac{B}{m},$$

so that "B" is the expected number of batches on station consisting of parallel BM and "X" is the expected number of batches on each machine. Values of θ are set according to the variable "X" and number of jobs. Finally, for each configuration of n, m, p_j, s_j, β, θ, and D_{PM}, there are 10 different problems, which result in a total set of experiments of 5400 instances which each experiment runs six times. In Policy III, four parameters exist: θ, β, $R_0(t)$, and t. The same configurations of β, θ, and D_{PM} as in the case of policy II are considered; therefore, a set of 5400 instances is also obtained. On the other hand, the aim of policy III is to keep a minimum level of reliability for a production period t which T_{PM} is calculated by Eq. (2). Here, the aim is a 95% reliability after the production period t, thus $R_0(t)$=0.95. Thereupon, it is necessary to determine period t to be calculated T_{PM} by Eq. (2). Parameter t can be easily obtained from the job processing times of the instances. Processing times are generated from uniform distributions between [1,10] and [1,20], and then, $t \approx X \times 5$ and $t \approx X \times 10$, respectively. After calculating B and then X, we show values of t in Tables 2 and 3.

Tab. 2. Values of t for m=2

problems	t				
	n = 10	n = 20	n = 50	n = 100	n = 200
$s_j \in [2,4]$, $p_j \in [1,10]$	15	24	37	155	160
$s_j \in [4,8]$, $p_j \in [1,10]$	16	36	84	167	341
$s_j \in [1,10]$, $p_j \in [1,10]$	12	27	77	148	324
$s_j \in [2,4]$, $p_j \in [1,20]$	25	30	75	321	373
$s_j \in [4,8]$, $p_j \in [1,20]$	33	73	169	335	683
$s_j \in [1,10]$, $p_j \in [1,20]$	25	55	155	296	648

Tab. 3. Values of t for m=4

problems	t				
	n = 10	n = 20	n = 50	n = 100	n = 200
$s_j \in [2,4]$, $p_j \in [1,10]$	13	15	28	80	93
$s_j \in [4,8]$, $p_j \in [1,10]$	13	16	42	97	170
$s_j \in [1,10]$, $p_j \in [1,10]$	15	27	38	74	162
$s_j \in [2,4]$, $p_j \in [1,20]$	13	17	37	160	186
$s_j \in [4,8]$, $p_j \in [1,20]$	16	36	84	167	341
$s_j \in [1,10]$, $p_j \in [1,20]$	13	27	79	143	324

Each category of problems is represented with a run code. For example, a problem category with 10 jobs, four BMs, processing times generated from the discrete uniform [1,20], job sizes generated from the discrete uniform [4,8], and β=3 is shown by *J*1*m*2*p*2*s*3*β*3. To be fair, searching time is set to a specific number for both algorithms, which is equal to 0.5×n×m milliseconds. This criterion is sensitive to both problem sizes (i.e., n and m). By using this stopping criterion, searching time increases according to the increase of either a number of DCs or customers. Algorithms are coded in C# (visual studio 2008) and run on a Pentium IV PC with 2.6 GHz and 1GB of RAM memory. With these data, the Taguchi experiments are carried out for each algorithm. The results are individually transformed to S/N Ratio, then these ratios of trials are averaged at each level.

5. Performance Assessment

In this section, we compare the results of algorithms. Because the scale of objective functions in each instance is not the same, the relative deviation percentage or response variable (RPD) is used as a common performance measure to compare the algorithms. After transforming the results of experiment to RPD measure, as illustrated in Tables 4 and 5, they are averaged for each combination of n and m (48 data points per average). This table shows the high performance of AIA (RPD=0.0155%) with respect to the GA. Figures 11 and 12 demonstrate interaction between the quality of the algorithms and the size of problems. It is worth noting that AIA exhibits the robust performance, meanwhile the problems of size increases. It also shows remarkable improvement in performance of AIA in large-sized problems versus GA.

Tab. 4. Average RPD for the algorithms grouped by n and m=2

	Algorithms	
n / m =2	GA	AIA
10	0.0286	0.0232
20	0.0279	0.0223
50	0.0352	0.017
100	0.0163	0.0088
200	0.0266	0.0064
Average	0.02692	0.01554

Tab. 5. Average RPD for the algorithms grouped by n and m=4

	Algorithms	
n / m=4	GA	AIA
10	0.032	0.0212
20	0.0279	0.0213
50	0.033	0.016
100	0.0163	0.015
200	0.0166	0.0054
Average	0.02516	0.01578

6. Conclusion

In advanced manufacturing systems, the main purpose of preventive maintenance is to ensure that all machines required for production are operating at high level of operating conditions at all times. Therefore, manufacturers should consider having an established maintenance schedule in place in order to reap the full benefits of a preventive maintenance strategy. In this paper, both fixed and flexible (*Non fixed*) operations are considered to achieve a plan for scheduled preventive maintenance at a manufacturing system with parallel machines. The *fixed PM* is considered where PM operations are planned in advance and the starting times of the maintenance operations are fixed beforehand. In this type of PM, the fixed time periods are predefined without considering probabilistic

models and PM operations are carried out exactly in these time periods. On the other hand, the schedule of PM periods is determined jointly with the schedule of jobs in *Non-fixed* PM, and the starting times of the PM activities are allowed to be flexible. In this type, the time of the PM activities is not determined precisely, but the times needed to perform PM activities times are computed based on the failure times of a machine. Two solution methodologies, GA and AIA, are devised to reach the appropriate scheduled maintenance plan due to time complexity of the problem. The best performance of methods is investigated according to their parameters and operations. Then, the performance of solution methodologies is evaluated via computational experiments.

References

[1] Cassady CR, Kutanoglu E. Minimizing job tardiness using integrated preventive maintenance planning and production scheduling, IIE Transactions, Vol. 35, (2003), pp. 503-513.

[2] Chang PY, Damodaran P, Melouk S. Minimizing makespan on parallel batch processing machines, International Journal of production research, Vol. 42, (2004), pp. 4211–4220.

[3] Dasgupta D, Ji Z, Gonzalez F. Artificial immune system (AIS) research in the last five years, in: Proceedings of the IEEE Congress on Evolutionary Computation, Canberra: Australia, (2003), pp. 123-130.

[4] Damodaran P, Velez-Gallego MC. Heuristics for makespan minimization on parallel batch processing machines with unequal job ready times, International Journal Advanced Manufacturing and Technology, (2009), DOI 10.1007/s00170-009-2457-1.

[5] Damodaran P, Vélez-Gallego M, Maya J. A GRASP approach for makespan minimization on parallel batch processing machines, Journal of Intelligent Manufacturing, (2009), DOI 10.1007/s10845-009-0272.

[6] Damodaran P, Vélez-Gallego MC. A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times, Expert Systems with Applications, Vol. 39, (2012), pp. 1451-1458.

[7] de Castro LN, Timmis J. Artificial immune systems as a novel soft computing paradigm, Soft Computing Journal, Vol. 7, (2003), pp. 526-544.

[8] de Castro LN, Timmis J. An artificial immune network for multimodal function optimization, Evolutionary Computation, CEC'02, in: Proceedings of the 2002 Congress, (2002a), pp. 699-704.

[9] de Castro LN, Timmis J. Artificial immune systems: A new computational approach, Springer, London. UK, (2002b).

[10] de Castro LN, Von Zuben FJ. Learning and optimization using the clonal selection principle, IEEE Transactions on Evolutionary Computation, Vol. 6, No. 3, (2002c), pp. 239-251.

[11] Engin O, Doyen A. A new approach to solve hybrid flow shop scheduling problems by artificial immune system. Future Generation Computer Systems, Vol. 20, (2004), pp. 1083-1095.

[12] Husseinzadeh Kashan A, Karimi B, Jenabi M. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes, Computers & Operations Research, Vol. 35, (2008), pp. 1084-1098.

[13] Liu LL, Ng CT, Cheng TCE. Scheduling jobs with release dates on parallel batch processing machines. Discrete Applied Mathematics, Vol. 157, (2009), pp. 1825-1830.

[14] Malve S, Uzsoy R. A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. Computers & Operations Research, Vol. 34, (2007), pp. 3016-3028.

[15] Mönch L, Balasubramanian H, Fowler JW, Pfund ME. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times, Computers

& Operations Research, Vol. 32, (2005), pp. 2731-2750.

[16] Ruiz R, García-Díaz JC, Maroto C. Considering scheduling and preventive maintenance in the flowshop sequencing problem, Computers and Operations Research, Vol. 34, (2007), pp. 3314-3330.

[17] Sortrakul N, Nachtmann HL, Cassady CR. Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single Machine, Computers in Industry, Vol. 56, (2005), pp. 161-168.

[18] Wang HM, Chou FD. Solving the parallel batch-processing machines with different release times, job sizes, and capacity limits by metaheuristics, Expert Systems with Applications, Vol. 37, (2010), pp. 1510-1521.

[19] Yazdani Sabouni MT, Jolai F, Mansouri A. 2008, Heuristics for minimizing total completion time and maximum lateness on identical parallel machines with setup times. Journal of Intelligent Manufacturing, DOI 10.1007/s10845-008-0191-4.

[20] Jamshidi R, Seyyed Esfahani MM. Human resources scheduling based on machines maintenance planning and human reliability level, International Journal of Industrial Engineering & Production Research, Vol. 6, No. 1, (2015), pp. 27-38.

[21] Jalali Naini SG, Aryanezhad MB, Jabbarzadeh A, Babaei H. Condition based maintenance for two-component systems with reliability and cost considerations, International Journal of Industrial Engineering & Production Research, Vol. 20, No. 3, (2009), pp. 107-116.

[22] Riahi M, Ansarifard M. Maintenance improvement of ball bearings for industrial applications, International Journal of Industrial Engineering & Production Research, Vol. 19, No. 7, (2008), pp. 123-126.

[23] Mokhtari H, Mozdgir A, Nakhai Kamal Abadi I. A reliability/availability approach to joint production and maintenance scheduling with multiple preventive maintenance services, International Journal of Production Research, Vol. 50, No. 20, (2012), pp. 5906-5925.

[24] Mokhtari H, Dadgar M. Scheduling optimization of a stochastic flexible job-shop system with time-varying machine failure rate, Computers & Operations Research, Vol. 61, (2015), pp. 31-45.